Large file processing with sed and AWK

version 2024

Christine Gaspin
Christophe Klopp
http://bioinfo.genotoul.fr/
http://www.sigenae.org/

Course organization

- 9:00 AM to 5:00 PM
- Lunch break 12:00 AM till 2:00 PM
- Five minutes breaks every hour 10:00 AM to 10:05 AM...
- The idea is to adapt to all participants. Therefore feel free to ask questions, to discuss with your neighbor, to check solutions with others, to search the Internet, to ask an AI…
- We will learn from your questions and our errors.

Do you have command line access?

- Use your local machine or a distant server
  - Local :
    - *nix or mac open a terminal
    - MS-Windows open mobaXterm
  - Distant : https://mobaxterm.mobatek.net/
    - *nix or mac open a terminal and connect to server using ssh
    - MS-Windows open putty or mobaXterm and connect to the server
- Paste the result of the date command in the chat followed by you name.

Self presentation

- Name
- Laboratory
- Research field or object
- What are you expecting from this day?

What you are going to learn about sed?

- understand and know how to explain the structure of a sed command
- find the function letter in the command
- make the difference in absolute and relative addressing
- read and understand a simple sed command
- write a sed command from simple specifications
- know the functions of the letters "d, a, p, i, s"

Questions

- What is a file?
- What is a directory?
- What is a process?
- What is Unix?
- What is a shell?

Sed command structure

sed -options 'address,range **command** command_parameters' file_name

command string

Examples :

```
sed '10d' myfile
sed '12,14d' myfile
sed '1ithis is the header' myfile
sed -n '^,100p' myfile
sed '/AA/,/^CC/s/DD/aa/g' myfile
sed -nr '/^[AB]{3}$/p' myfile
```

Regular expressions examples

```
a* : rien, a, aa, aaa, aaaa, ...(nombre infini de a)
a(ab)+ : aab, aabab, aababab, ...
[a-d] : a,b,c,d
[a-c]-[1-3] : a-1,b-1,c-1,a-2,b-2,...
^a(1de)+ : a line starting with a1de or a1de1de, ...
abde$ : a line ending with abde
file\.txt : file.txt
ab.d : abad, abbd, abcd, abdd, ...
[1-31]/[1-12]/[0-2000] : date (first method)... 11/10/1972
[lL]iberty : liberty, Liberty
[pP]age [12-99] : page 12, Page 12, ..., page 99, Page 99
```

^ = line start $ = line end

[] = character range [-] character ranges

*** = any number of characters**

() = character block ()+ any number of character blocks

. = one character

Regular expressions exercises

Find regular expressions which

1) Extract lines with the character 1
2) Extract lines with only two characters
3) Extract lines beginning with a bracket
4) Extract lines with only upper case letters

What are the following regular expression extracting ?

5) ^$
6) [a-z]+
7) ^[0-9]$
8) m[oai]ts
9) [a-zA-Z]{6}
10) [0-9]{2,4}
11) ^[ATGC]+$
12) p(ai|i)n

Invoking sed

```
> sed -n '/^>/p' myfile.fa
> cat myfile.fa | sed -n '/^>/p'
> sed -n '/^>/p' < myfile.fa
```

If filter.sed contains /^>/p then

```
> cat myfile.fa | sed -n -f filter.sed
> sed -n -f filter.sed myfile.fa
> sed -n -f filter.sed < myfile.fa
```

If you have a large number of sed commands use a file.

sed options

sed command options = before the quotes

parameter	description
-n, --quiet, --silent	suppress automatic printing of pattern space
-e script, --expression=script	add the script to the commands to be executed
-f script-file, --file=script-file	add the contents of script-file to the commands to be executed
--follow-symlinks	follow symlinks when processing in place
-i[SUFFIX], --in-place[=SUFFIX]	edit files in place (makes backup if SUFFIX supplied)
-s, --separate	consider files as separate rather than as a single continuous long stream.

-r --regexp-extended / use extended regular expressions in the script.

Sed adresses

values	examples
Line numbers	1 2 12 $
Start and end line numbers	1,2 12,$ 1,6
pattern	/AA/ /^AA/ /AA$/
Pattern ranges	/^A/,/C$/ /X/,/Y/
combinations	/AA/,$ 1,/^AA/

You can reverte a selection by adding !

Examples :

```
sed -n '/AA/!p' file
sed -n '1,4!p' file
```

sed multiple commands

sed 'command_string1; c_s2;…; c_sn' input_file

sed -e 'command_string1' -e 'c_s2' -e 'c_s3' input_file

sed 'command_string1' input_file | sed 'c_s2' | sed 'c_s3'

Examples :

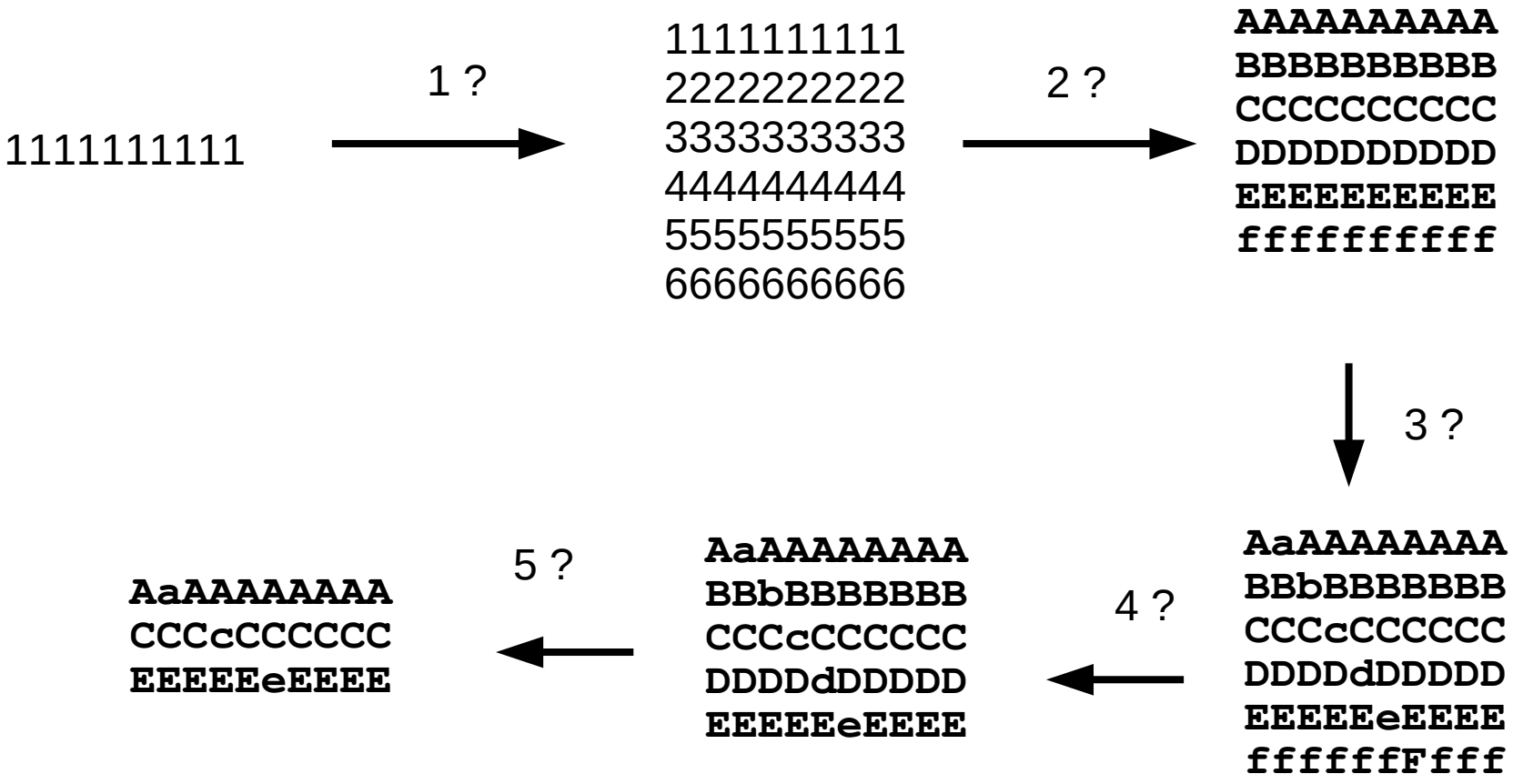
```
sed '10d;15d;17d' file

sed –e '10d' –e '15d' –e '17d' file

sed '10d' file | sed '15d' | sed '17d'
```

File modification exercise

- Create file test.txt : echo '1111111111' > test



Data files used for the execises

- Fasta : sequence file
- GTF : gene transfer format
- VCF : variant call format
- Expression measurements : text file with counts

- Files have headers and content
- File can be structured in columns

- http://genoweb.toulouse.inra.fr/~klopp/SedAwk2021/Data/

Exercise 1 : simple file processing

- Download data in your local directory
- Unzip files

- Find header lines format (unix commands)

- Remove header from each file. How many lines are remaining?
- Remove non header lines in the fasta file. How many sequences?
- Filter gene lines in the gff3 file. How many genes?
- Rename contigs by adding CONTIG_ after ">" in the fasta file.

Exercise 2

- What is the sed command to extract the LOC100698896_ORENI.1.4 sequence from the contig file?
- List files from your directory and build a unix command file to count the number of lines of each file.
- Compute the line number of the VCF file after having removed the header part.

sed conclusions

- sed is a powerful file processing tool.
- It enables to remove, add modify lines
- The filtering can be very precise using regular expression
- It is very quick processing millions of lines per minute

What you are going to learn about awk?

- understand and build a simple one line awk command
- know and know how to use the awk built-in variable
- be able to format a printed character string
- create and set a variable
- create and set an array
- use the BEGIN and END statements

AWK command structure

```
awk 'awk command string' filename.txt
```

awk command string structure :

'**BEGIN**{statements} *FILTER*{statements}**END**{ statements}'

- **BEGIN** runs before reading the input file
- *FILTER* RE to select the processed lines
- **END** runs after reading the input file
- Statements are separated by « ; »

Use simple or double quotes at command ends.

awk command examples

```
awk '/^AA/' mon_fichier.txt
awk '$2 == 5' mon_fichier.txt
awk '$1 == 5 || $8 == 12' mon_fichier.txt
awk '$3 == 6 {print $7"\t"$9;}' mon_fichier.txt
awk 'BEGIN{print "test";}'
awk '{print "test";}' mon_fichier.txt
awk 'END{print NR;}' mon_fichier.txt
```

AWK built-in variables

Built-in variables

`$0` `$1, $2 … $NF`	Whole line, first, second… last field
`ARGC`	Number of command line arguments
`ARGV`	Array of command line arguments
`FILENAME`	Name of current input file
`FS, RS`	Input field / record separator (def: one space, \n)
`NF`	Number of fields in current record
`NR, FNR`	Number of record read so far / in current file
`OFMT`	Output format for numbers (default: %.6g)
`OFS, ORS`	Output field / rec. separator (def: one space, \n)
`RESTART,` `RLENGTH`	Start / Length of string matched by `match` function (see below)
`SUBSEP`	Subscript separator (default: \034)

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

awk print command

```
awk '{print ”a”}' myfyle.txt
awk '{print a}' myfyle.txt
awk 'END{print NR;}' myfyle.txt
awk '{print $3”\t”$2”;”$1;}' myfyle.txt
awk 'BEGIN{OFS=”\t”}{print $1,$2,$3;}' myfyle.txt
```

What are the differences between :

```
awk 'BEGIN{a="A"; b="B"; print ab}'
awk 'BEGIN{a="A"; b="B"; print a""b}'
awk 'BEGIN{a="A"; b="B"; print a"\t"b}'
awk 'BEGIN{a="A"; b="B"; print a","b}'
awk 'BEGIN{a="A"; b="B"; print a,b}'
```

Print and built-in variable exercises

test.txt content

```
AaAAAAAAAA
CCCcCCCCCC
EEEEEeEEEE
```

1) Using the test.txt input file write an awk command printing 4 times the content per line separated by semi-columns (;)

2) Using the test.txt input file write an awk command printing the line number, line content, number of fields separated by tabs (\t)

3) add the file names at the first position of the output of the previous exercise.

AWK operators

Operators	
&& || !	Logical operators. Ex: !($2<4 || $3<20)
< <= == != >= >	Comparing operators
~ !~	matched by, not
selector?if-true-exp:if-false-exp	

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

Question

If your current directory contains the following files

```
Texte1.pdf
Textefasta2.pdf
data1.fa
data2.fa
```

What produces the following command ?

```
ls | awk 'BEGIN { print "fasta in this directory
are:" } /.fa$/ { print } END { print "Done!" }'
```

Awk exercises 1

Using the following file
http://genoweb.toulouse.inra.fr/~klopp/SedAwk2021/Data/merged_count.csv.gz

- Extract lines finishing with 2.
- Extract lines from 10 to 20.
- Extract column 5 and 7 from the count file.
- Extract column 5 and 7 if value in column 6 greater than 1000.
- Extract lines where second column is longer than 13 (use the string length function : length()).

Awk exercises 2

- Create a new file with column 1,2 and value of column 3 multiplied by 3.
- Create a new file with line numbers in the first column and the number of fields in the last one.
- Create a new file with the sum of the line at the end of the line.
- Calculate sum of column 4 when column value is less then 100.

AWK functions

Main built-in functions

r: regex ; s,t: strings ; n,p: integers

int(n), **sqrt**(n), **exp**(n), **log**(n), **sin**(n), **cos**(n)	
rand()	Random number between 0 and 1
close(file or command)	
getline [var] **getline** [var] < file command | **getline** [var]	Read next line from input file, from a specific file, or from a pipe Return 1 (record found), 0 (end of file), -1 (error)
gsub(r,s) **gsub**(r,s,t)	Substitute s for r globally in $0 / string t; return # of subs made
index(s,t)	Return first position of string t in s, or 0 if t is not present
length(s)	Return number of characters in s
match(s,r)	Test whether s contains a substring matched by r; return index or 0; sets RSTART and RLENGTH
split(s,a) **split**(s,a,fs)	Split s into array a on FS / field separaror fs; return # of fields
sprintf(fmt,expr-list)	Return expr-list formatted according to format string fmt
sub(r,s) **sub**(r,s,t)	Substitute s for the leftmost longest substring of $0 / t matched by r; return # of subs made
substr(s,p) **substr**(s,p,n)	Return substring of s (of length n) starting at position p
tolower(s), **toupper**(s)	Lower and upper cases

http://www.cheat-sheets.org/saved-copy/awk_quickref.pdf

Awk exercises 3

http://genoweb.toulouse.inra.fr/~klopp/SedAwk2021/Data/contigs.fasta.gz

- Transform fasta file in tab file with first column corresponding to file name and second column corresponding to sequence

- Add sequence number to the previous file

- Add sequence length to the previous file

http://genoweb.toulouse.inra.fr/~klopp/SedAwk2021/Data/snpdetect.vcf.gz

- Extract genotypes from VCF file

awk conclusions

- awk is a powerful file processing tool.
- Built-in variable simplify scripting
- awk has a many functions which can be used to filter or aggregate data
- awk is adapted to process file columns

Survey link

https://sondages.inrae.fr/index.php/84236?lang=fr